Amendments to the Specification

Please replace the paragraph on Page 1, lines 5 - 7 with the following marked-up replacement paragraph:

-- The present invention is related to commonly-assigned U. S. Patent _____ (serial ~~number 09/_____), which~~ number 09/973,883), which is titled "Adaptive Indexing Technique for Use with Electronic Objects" and which was filed concurrently herewith. --

Please replace paragraph that begins on Page 5, line 7 and carries over to Page 6, line 2 with the following marked-up replacement paragraph:

-- Messages can be transferred from one mail folder to another manually, using a drag-and-drop operation. In addition, in many systems, the user can define a mailbox "filter" which transfers messages from one ~~inbox~~ mailbox to another (equivalently, from one folder to another). A filter operates as a type of macro, with the underlying programmatic support provided by the e-mail system in response to the user's specification of filter parameters. For example, criteria for invoking a transfer from a user's incoming mailbox to the trash folder might include factors such as the presence of a particular character string in the body of the mail message or in the "From" field or "Subject" field, and so forth. Upon arrival of an incoming message, it is compared to the defined filters to determine if any type of transfer should be performed. If the message matches the defined filtering criteria, then an automatic transfer may be performed. Filters can also be set to operate upon explicit request from the user (for example, by pressing a "Filter Messages" button), and may be defined to operate on folders other than the in-basket. Filters may also

perform a limited number of actions other than transferring messages, depending on the support provided by the e-mail system. For example, when using the Eudora system, filters can be used to dynamically change the priority of a message, and a particular filter may perform as many as five actions which are selected from the available actions list. --

Please replace the paragraph that begins on Page 28, line 9 and carries over to Page 29, line 20 with the following marked-up replacement paragraph:

-- Using the approach shown in Fig. 4, the user may indicate his desire to alter the organizing rules by (for example) pressing a function key or "right-clicking" with his mouse device or otherwise choosing this operation. This preferably causes a window such as pop-up window 400 to be displayed, offering several choices to the user such as viewing the contents of a node, using a visual rule builder (which has been selected in the example of Fig. 4), expanding or collapsing a node, and so forth. After choosing to use the rule builder, another window 410 is preferably displayed, where the contents of the window preferably include alternative categories that can be used for organizing the node's contents. (These alternative categories are preferably defined earlier by the user or provided as built-in defaults and deployed at the time of installing the implementation of the invention.) Suppose the user chooses "Design" as a criteria for organizing this node's content, as indicated by the visual shading and checked box in window 410. Another level of pop-up window, shown as window 420, appears next in this example, where window 420 allows the user to choose either to customize this category or to select the default (i.e. already-defined) customization for this category. If the user chooses "default", then it

is not necessary to display any subsequent pop-up windows, and the default organizing criteria will apply for node 225; otherwise, when the user chooses to customize this category, another pop-up window appears. In the example of Fig. 4, pop-up window 430 is displayed to enable the user to quickly locate his already-defined index criteria using an alphabetical index menu to easily access the criteria in alphabetical order. For example, to see the index terms beginning with the letter "D", the user selects "D" from pop-up window 430. The corresponding index terms are then displayed in another pop-up ~~window 450, as~~ window 440, as illustrated, enabling the user to select one or more of the defined terms or organization criteria. In the example, the user has selected a term "Design charts" and another term "Design messages". (In an actual tree view, these terms correspond to the node for which organizing rules are being customized.) In addition to (or instead of) displaying an alphabetic index representation in window 430, thumbnail sketches or other forms of non-text representations might be used. For example, a miniature version of the Mustang logo might be presented in window 430 to enable the user to select the Mustang image as an index criterion. A representation might also be included for grouping index terms which are not easily represented using other approaches, for example by including a grouping such as "Miscellaneous". After completing the selection process in window 440, the user might return to ~~window 410 to~~ window 430 to select additional criteria for use in organizing this node. (Note that the layout of windows 400 - 440, where one window appears to the right and slightly beneath its predecessor, is for purposes of illustration only; other presentation styles may be used alternatively.) --

Please replace the paragraph that begins on Page 29, line 21 and carries over to Page 31, line 5 with the following marked-up replacement paragraph:

-- The flowcharts in Figs. 5 - 7 illustrate logic which may be used to implement a preferred embodiment of one aspect of the present invention. The processing of this aspect, which comprises creating the relational object view which has been described, operates in response to particular events. The logic of Fig. 5 is preferably invoked when the user opens the tree view hierarchy, and the logic constructs and renders the object view. In preferred embodiments, this logic is also invoked (see Block 625 of Fig. 6) when the user requests to expand a node from the currently-displayed relational view, and the logic constructs and renders a subtree of that node. Fig. 5 may also be invoked when a new object arrives, and/or when new organizing criteria are defined, and in such cases creates a refreshed view. (Other triggering events might be defined for invoking Fig. 5, such as expiration of a refresh timer, if desired.) As shown therein, the process begins in Block 500 where all the ~~newly-arrived or newly-created~~ objects to be organized are retrieved (for example, from the "InObjects" object repository). If desired in a particular implementation of the present invention, the InObject repository may also store objects (that is, pointers to such objects) which on previous iterations did not meet any of the organizing criteria for placing them into the hierarchy. Those objects may also be retrieved and (re)processed against the currently-defined criteria, if desired. The objects may be of a variety of types, including e-mail messages, bitmaps, textual documents, and so forth. (As stated with reference to node 208 of Fig. 2, indications of the new and the not-categorized objects are preferably buffered for updating the tree view in a controlled manner.) Block 505 retrieves the

organizing criteria, which in preferred embodiments are stored as rules in a rules repository. These rules may include default rules as well as user-defined rules. (See the discussion of Fig. 7, below, for more information on defining these rules.) An object hierarchy agent process is then invoked (Block 510) to apply the criteria to the objects to be organized. Block 515 checks to see if there are any existing (i.e. already-indexed) objects to be merged with these objects. If so, then the merge proceeds (Block 520) by invoking a relational modeler process which creates an aggregated relational model of the user's objects. Finally, the organized objects are rendered in the relational view (Block 525). Preferably, a process such as a presentation manager component is invoked to perform the actual rendering of the constructed hierarchy. Presentation manager components are well known in the art, and are readily available. --

Please replace the paragraph on Page 33, lines 4 - 12 with the following marked-up replacement paragraph:

-- Block 750 formats a rule from the criteria which the user has indicated through this visual rule building process, and stores that rule in the rules base. If the user wishes to continue defining rules for the current node in this manner (a ~~positive~~ negative result for the test in Block 755), then the appropriate menu regains control (Block 760) and the process is repeated. (That is, if the user wishes to define more organizing criteria using a different letter of the alphabet via window 430, then control returns to Block 730; alternatively, the user may wish to return to the menu in window 410 or 420, in which case control returns to Block 710 or 720, ~~respectively. Or,~~ the respectively.) Or, the user may select a different node from the displayed hierarchical view

and define rules for that node by returning to Block 700. --

Please replace the paragraph on Page 38, lines 4 - 11 with the following marked-up replacement paragraph:

-- In this manner, the system according to the present invention can dynamically learn new organizing criteria in a very user-friendly (and user-customizable) manner. This technique may be used with swiping-capable devices that are considered "conventional" in the prior art, such as mouse devices, light pens, and plasma panels, and it may also be useful with devices or mechanisms which have not yet been imagined. The criteria selected using the swiping technique ~~is fed~~ are fed back into the system, as illustrated graphically in Fig. 8 (see elements 810 - 825, for example), and may be presented to the user for building rules to be used by an indexing engine (see, for example, the discussion of Fig. 4). --

Please replace the paragraph that begins on Page 38, line 13 and carries over to Page 39, line 7 with the following marked-up replacement paragraph:

-- As has been demonstrated, the present invention provides a number of advantages over the prior art. Whereas prior art e-mail systems allow users to sort e-mail messages on fixed message attributes (and, in some systems, on the presence of user-defined keywords within message bodies), and then display all the messages of a folder or mailbox in a flat view using this set of fields for sorting, the present invention enables a very flexible, dynamically-constructed hierarchical view (or other structured view) of an aggregation of electronic objects which is

organized using a user-specified, multi-level index. The disclosed techniques provide enhanced graphical presentation of information, giving an intuitive view of relationships among objects, and ~~allow~~ enable improved access to stored information. Using the disclosed techniques, each individual user can control how his or her information is organized, rather than being limited to choices which the developers of the e-mail system have chosen to expose. In addition, the multi-level indexing technique of the present invention enables a tree view to change dynamically, as new objects are received which meet active organization criteria, and allows node-specific organizing criteria for an arbitrarily complex level of nesting. This is in contrast to the static views provided by prior art systems (where the attributes are either in use for all messages arriving into the current mailbox or folder, or they are not in use for any messages of the mailbox or folder). --